METHOD AND APPARATUS FOR LAUNCHING COMPUTER APPLICATIONS

BACKGROUND

Field of the Invention

This invention is related to the way applications are started on a computer system. More particularly, this invention is related to starting applications on a computer from within a web browser.

Description of the Problem

Computer applications programmed in languages such as Visual Basic and C++ have the ability to interact with the host computer's operating system, including the file system. This ability allows those applications to easily locate and start other applications. World Wide Web applications, usually written in some form of hypertext markup language (HTML) however, do not share this same ability to interact with the host operating system. While Microsoft's "ActiveX" controls and scripting languages such as Visual Basic Script support interaction with the host operating system, these approaches are only supported on Microsoft's Windows™ based operating systems. JavaScript and Java Applets do not allow access to the Microsoft Window's registry which is where most application information is stored on Microsoft Windows based operating systems. What is needed is an efficient, fast and user-friendly way to identify, display, and launch applications within a computer without being tied down to a specific operating systems, and without unnecessary or distracting dialog being presented to a user.

SUMMARY

The present invention meets the above needs by providing a "launcher" application that operates on a computer. The launcher allows a user to quickly determine which applications are installed and to start one or more selected applications. The launcher presents information to the user using hypertext markup language (HTML), a standard scripting language that can be read by a standard web browser application, such as Netscape Navigator™ or Microsoft Internet Explorer™. Thus, a user is presented with a standard graphical user interface that he or she most likely knows. When we refer to the scripting language HTML in this disclosure, we mean HTML or any of its variants, such as dynamic HTML (DHTML). More advanced features, such as "graying out" of applications that are not installed are performed using more advanced, known, scripting languages such as ECMAScript. The launcher suppresses browser dialog boxes that a user may find out of place or inappropriate in the particular operating environment.

In one embodiment of the invention, the launcher accesses an operating system registry to determine which applications are installed on the computer and where each application is installed. The launcher then creates a hypertext markup language (HTML) file specifying the applications that are installed. The information in the HTML file is displayed so that a user can select any one of the applications that is installed as a selected application to launch. The user makes the selection in this embodiment by clicking with his or her mouse. The display may simply list the installed applications, or it may list all known applications, showing unavailable or uninstalled applications with a different visual attribute than installed applications, for

example, dimmed or grayed out text. If the user selects an application, the launcher determines which application was selected from tags in the hyperlink, stops browser navigation and starts the selected application.

In one embodiment, the invention works on a small computer such as a personal computer or workstation. Software that implements aspects of the present invention can be stored on a media. The invention is implemented by a computer program product, which includes a computer program containing computer instructions. The computer program product can be supplied on a media such as diskette, tape, or fixed disc, or optical, such as a CD-ROM. Additionally, the computer program product can be supplied via the Internet or some other type of network. Workstations or servers that run the software include a plurality of input/output devices, a processor, and memory devices that store and execute the instructions necessary to implement the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a flowchart that illustrates the method of one embodiment of the invention.

FIG. 2 shows a launcherflags.ini initialization file according to one embodiment of the invention.

FIG. 3 illustrates a launcherflags.htm HTML file according to one embodiment of the invention.

FIG. 4 shows a launcher.htm file according to one embodiment of the invention.

FIG. 5 illustrates a launchermenu.htm file according to one embodiment of the invention. FIG. 5 is divided into Figures 5-A through 5-J or convenience.

FIG. 6 shows the help text that can be used in the launchermenu.htm file of FIG. 5. FIG. 6 is divided into Figures 6-A through 6-D for convenience.

FIG. 7 is a block diagram illustrating an example computing system that executes the computer program product embodying the invention.

DETAILED DESCRIPTION OF ONE OR MORE EMBODIMENTS

FIG. 1 illustrates the overall method of the invention according to one embodiment. The launcher is an application that utilizes the Microsoft™ WebBrowser control, which is the foundation for the Microsoft™ Internet Explorer application. The WebBrowser control is a Windows™ extension that is capable of displaying an HTML page, as you would see it in a web browser.

It is highly desirable for the invention to use a standard interface that works with many applications. A web browser interface is a good choice since it is well known because of the popularity of the Internet.

In one embodiment, when the launcher is started, it checks the Windows registry at step 101 of FIG. 1 to determine which applications are installed. It then reads an initialization file, commonly known as an "ini" file. In this embodiment, the file is called LauncherFlags.ini (see FIG. 2), the launcher writes out a copy of it in HTML at step 102. In this embodiment, the HTML file is named LauncherFlags.htm (see FIG. 3). To make the copy, the contents of LauncherFlags.ini are copied line by line and lines containing the text "tkn=" are updated to reflect the status of that element on the

PC or workstation running the launcher. Each application has its own line and hence its own "tkn=" value. For applications that are not installed, the "path=" portion of the line will be made equal to an empty value, in this example, empty double quotes. For applications that are installed, the "path =" portion of the line will be made equal to a unique name followed by the text ".tag". After LauncherFlags.htm has been written, an HTML portion of the launcher, called Launcher.htm, then asks the WebBrowser control to display an HTML page (see FIG. 4). The appropriate HTML file or files is displayed in a window at step 103 so that a user can select an application to launch.

In this example, Launcher.htm loads two frames at step 103. Launcher-Flags.htm is in one frame and that entire frame is made invisible, and Launcher-Menu.htm (See FIG. 5) is in a second frame and that frame is made visible. DHTML and ECMAScript in LauncherMenu.htm reference the id's in LauncherFlags.htm to determine what color the text should be for each of the launchable applications. Dynamic HTML is an extension to HTML that allows dynamic positioning, hiding and showing of HTML elements. This color distinction provides a visual attribute to distinguish installed from uninstalled or unavailable applications (black text if the application is installed, grayed out text if the application is not installed). When the user mouse clicks on an application name that is installed, the selection is detected at step 104, and LauncherMenu.htm tries to navigate to whatever was specified in the "path portion of the matching id in LauncherFlags.htm. The WebBrowser control allows the launcher to "see" where the user is trying to navigate and to optionally stop that navigation. The launcher application stops all navigation by the browser to paths ending in ".tag" and reads the unique text that preceded the ".tag" extension and does a look

up with that text to determine which application the user wants to start and where it is located at step 105. Standard code for starting another process is then used to start the user-requested application at step 106. Since the browser does not start the application, the user does not see a disruptive dialog box asking whether the user wants to save or run the application, or warning the user that an application is about to be run.

Figures 2, 3, 4, and 5 illustrate examples of files used in one embodiment of the invention. FIG. 2 is an example LauncherFlags.ini file. Each time the launcher starts, it copies this file to LauncherFlags.htm, but inserts values for the paths for applications that are installed. Notice that this file shows multiple applications. Names that have been used for applications by way of example include "mentor", "linkview", "dominocore", "examine", "atmapp", "frping", "console", and others listed in the "span" statements. Also note that all paths are equal to nothing. "SPAN" is an HTML tag that is not displayed and thus is used to define the "tkn", "id" and "path" variables that are used by LauncherMenu.htm. FIG. 3 shows the result of LauncherFlags.ini being copied to LauncherFlags.htm. Each application that is installed has a value for a path. For example, the application "mentor" has a path of "mentor.tag". In FIG. 3, paths are null for applications "atmapp", "console", "wizard", "toolbox", "ipfilter", and "dna323". FIG. 4 shows the HTML portion of the launcher application. The "frameset rows" statement hides the frame containing LauncherFlags.htm. Only LauncherMenu.htm is visible.

FIG. 5 illustrates an example of a LauncherMenu.htm file. FIG. 5 is divided into Figures 5-A through 5-J for convenience. Note that much of the HTML code

generates displayed text. It is convenient to be able to translate this text into different languages to facilitate ease of use on computer systems of various countries. Sections of the file that can be translated are commented. The file includes a space where a translation help file can be inserted near the end. FIG. 6 illustrates an example of such a help file. FIG. 6 is divided into Figures 6-A through 6-D for convenience. The files illustrated in Figures 5 and 6 refer to a particular product of the assignee of the invention called "DominoNAS." However the various HTML statements that generate title and other text information can be modified to refer to any product or software program that might accompany a launcher according to the present invention. Applications referred to in the Figures include, as before, PCAnywhere, Mentor, and others, but the invention can be made to work with any applications.

As previously mentioned, much of the software that is used to implement the invention resides on and runs on a computer system, which in one embodiment, is a personal computer, workstation, or server. FIG. 7 illustrates further detail of a computer system that is implementing the invention. System bus 701 interconnects the major components. The system is controlled by microprocessor 702, which serves as the central processing unit (CPU) for the system. System memory 703 is typically divided into multiple types of memory or memory areas, such as read-only memory (ROM), random-access memory (RAM) and others. If the computer system is an IBM compatible personal computer, the system memory also contains a basic input/output system (BIOS). A plurality of general input/output (I/O) adapters or devices, 704, is present. Only two are shown for simplicity. These connect to various

devices including a fixed disk, 705, a diskette drive, 706, and a display, 707. The computer program instructions for implementing the invention are stored on the fixed disk, 705, as a computer program product and are partially loaded into memory 703 and executed by microprocessor 702. The system also includes another I/O device, a network adapter or modem, shown at 708, for connection to the network, 709. It should be noted that the system as shown in FIG. 7 is meant as an illustrative example only. Numerous types of general-purpose computer systems are available and can be used to implement the invention. Available systems include those that run operating systems such as Windows™ by Microsoft and various versions of UNIX.

As previously mentioned, a computer program product in combination with the appropriate hardware implements the invention. This computer program product includes a computer program made up of computer program code or instructions. The computer program code is often stored on storage media. This media can be a diskette, hard disk, CD-ROM, DVD-ROM, or tape. The media can also be a memory storage device or collection of memory storage devices such a read-only memory (ROM) or random access memory (RAM). Additionally, the computer program code can be transferred to a workstation over the Internet or some other type of network. The diskette drive of FIG. 7 is indicated by a drawing of one type of media, a diskette, which can be used to initially transfer some of the computer program code of the invention to the computer system of FIG. 7. A diskette typically includes magnetic media enclosed in a protective jacket. Magnetic field changes over the surface of the magnetic media are used to encode the computer program code.

We have described specific embodiments of our invention, which provides a way to start applications from a web browser on a computer. One of ordinary skill in the computer and networking arts will quickly recognize that the invention has numerous other embodiments. In fact, many implementations are possible. The following claims are in no way intended to limit the scope of the invention to the specific embodiments described.

We claim: